Reliable causal delivery with probabilistic design

Jordi Martori , Pascal Urso

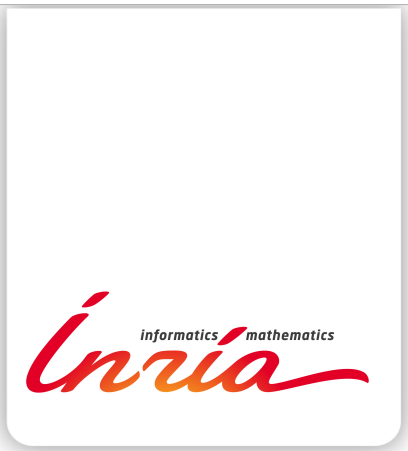
RESEARCH REPORT

N° 8985

November 2016

Project-Teams COAST Team

ISSN 0249-6399 ISRN INRIA/RR--8985--FR+ENG



Reliable causal delivery with probabilistic design

Jordi Martori *, Pascal Urso †

Project-Teams COAST Team

Research Report n° 8985 — version release 1.0 — initial version
November 2016 — revised version November 2016 — 15 pages

Abstract: Ensuring reliable and ordered communication between computers usually requires acknowledgment messages. In systems with a high rate of broadcast communication, the cost of such acknowledgment messages can be large. We propose to use the causal ordering information required by some applications to detect and request missing messages. To circumscribe the number of unnecessary requests we combine local awareness and probabilistic methods. Our model allow to obtain reliable communication within a latency equivalent to unordered communication and lower network usage than acknowledgment systems.

Key-words: Causality, Latency, Probability, Clocks, Reliability

* Université de Lorrraine – COAST Team - Inria, Loria, Nancy, France
† Université de Lorrraine – COAST Team - Inria, Loria, Nancy, France

RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Délivrance causale fiable basée sur les probabilités

Résumé : Assurer une communication ordonnée et fiable entre ordinateurs requière usuellement l'utilisation de messages d'accusé de réception. Dans les systèmes ayant un rythme élevé de communication d'ensemble, la charge de ces accusés de réception sur le réseau peut-être importante. Nous proposons d'utiliser les méta-données permettant l'ordonnancement causal pour détecter et récupérer les messages perdus. Afin de limiter le nombre de récupération inutiles nous combinons une connaissance locale du comportement du système ainsi que des méthodes probabilistes. Notre modèle nous permet d'obtenir une communication fiable avec des latences équivalente à une communication non-ordonnée et une charge réseau plus faible que les systèmes classique d'accusé de réception.

Mots-clés : Causalité, Latence, Probabilités, Horloges, Fiabilité

1 Introduction

Modern distributed systems need to be fast, available, scalable and consistent. Georeplication solves the first two issues, as the clients can contact a replica close to them. In case of failure, other replicas are still accessible and the client can continue its work with them. As stated by the CAP theorem, a system cannot provide linear consistency while being available and tolerant to partition, however it can provide eventual consistency. Eventual consistency states that if all errors are transient and there are no more updates, the system will eventually converge to the same state.

Partial orders provide a consistent view of an object across the whole system. All dependent events on an object need to be applied in the same order in all the machines. While such orders provides a more manageable environment for developers to work and ensures that users will not see unexpected behaviours, enforcing partial orders has an impact of the overall message delivery latency.

In this work, we study partial order enforcement impact on message delivery latency. We create several models that describe the behavior of the send messages under different latency models. This allows us to compute the probability that a message will be deliverable in one, many or all the nodes in the system. We currently present the models for FIFO and causal partial orders with exponential distribution as latency model; normal, uniform or exponential times between the events; and a fair repartition of the workload across all the nodes in the system.

The rest of the article is structured as follows:
Section 2 provides a brief description of the article's context. Section 4 introduces the theoretical work and the equations to compute the probabilities. Section 6 details summarizes the related work to this article, and the conclusions are drawn in Section 7.

2 Background

In this section, we provide background on partial orders and probabilistic latency models, they will be relevant for Section 4.

Between distributed systems, the need to causally broadcast every event to every node is particularly important in massively replicated data stores [6, 12] and collaborative systems [20, 24]. Moreover, in these systems, since the data is replicated and mutable, a consistency model must be specified.

2.1 Partial Orders

In what follows, the terms *event* and *message*. An event occurs first on a node, called *original node*, and it is sent within a message to other nodes to be eventually delivered. So, each message contains an event, and both terms can be used as synonyms when there is no confusion.

Enforcing an order between events across various nodes of a system allows the developers to work in an environment that behaves more naturally [13]. Events that happened before are applied before. This avoids some unexpected behaviors that the users could experience, for instance, the reappearance of a deleted file, or reverse order of question answer in a chat.

In distributed systems, the order between events can be total or partial. If the system enforces an order, when a message is received, the contained event might not be delivered immediately. The system delivers the event only when all the preceding events – according to the order – have been delivered.

If the order is total, all events are applied in the same order on all nodes. Such an order allows obtaining sequential consistency (or linearizability) of replicated data. However, the CAP theorem [8] states that such a consistency level is impossible to achieve in an available distributed system subject to partitions.

Partial orderings allow data congruency, as all nodes see in the same order the events that have dependencies while the concurrent ones can be seen in different orders. Under a partial order, all events that happened before have to be delivered before events that happened after. Depending on the partial order that is going to be enforced, the "happened before" definition is different. In FIFO partial order, an event A happened before B if both A and B belong to the same node and if A occurred before B in real-time.

Causal order [13], extends the definition of happened before of the FIFO ordering with: (1) an event A happened before B if they belong to different nodes and A was delivered to the node were B comes from before B was issued. And (2) if event A happened before B and B happened before C, then A also happened before C (transitivity).

2.2 Latency Models

Latency models allow to probabilistically represent the behaviors of communication in the system. Different latency models can be defined to represent a different kind of behaviors. The probability that a message takes less than a time t to be received is defined using a cumulative distribution function $P(X \leq t)$.

To simplify presentation, the latency model that we use for our system description is an exponential distribution. In Equation 1, λ^{-1} represents the mean time of communication, and t the time since issuance of the event.

$$P(X \leq t) = 1 - e^{-\lambda t} \tag{1}$$

An other latency model is a mixed type distribution between pareto and exponential distributions. It models production-like latency scenarios from Internet scale enterprises [2]. The pareto distribution models the bulk latency time and the exponential one models the tail. Equation 2 describes this model where α is the shape and x_m is the scale of the pareto distribution; and where λ^{-1} is the mean value of the exponential distribution, while p is the proportion of exponential distribution.

$$P(X \leq t) = p(1 - e^{-\lambda t}) + (1 - p)(\frac{\alpha x_m^{\alpha}}{t^{\alpha+1}}) \tag{2}$$

3 Model

For all the equations in Section 4, and unless expressed differently, we will be following the next assumptions. First, the number of nodes, n, in the system does not change. Second, the workload fairly shared amongst all the nodes. Third, the rate of operations remains constant, and the rate frequency is the mean time between operations being send can follow a uniform, exponential or

normal distribution. Fourth, the latency model's configuration does not change. Fifth, as in eventually consistency systems, every message is eventually delivered. However the latency models used represent the loss of a message with an arbitrarily long time[1]. Finally, all send messages are independent from one another.

4 Probabilistic Partial Orders

In this section, we introduce our first contribution which is probabilistic models for delivery latency in distributed systems enforcing a partial order. We take into account two different partial orders: FIFO and causal. The first set of equations define the delivery of one particular event, later we generalize these equations to determine the delivery of any message.

4.1 FIFO

With a FIFO partial order constraint, all previous messages from the node that sent the message need to be received before a newer message can be delivered. Equation 3 expresses the probability that the k-th message that a node has sent is ready to be delivered to another node. Since all messages being sent are independent, This probability is the product of the probabilities of reception of this message and all previous messages. δ_i is the time difference between the k-th and i-th event ($\delta_k = 0$), and $P(X_i \leq t)$ is the latency model of our system.

$$P(F_k \leq t) = \prod^{i \leq k} P(X_i \leq t + \delta_i) \tag{3}$$

4.2 Causal

The second partial order that we present is the causal order. Recall that the causal order arranges received and sent messages. So we need to distinguish, for a given node, *local* events that are originally issued on this node from *remote* events that are sent and delivered to this node.

Equation 4 gives us the probability that a local event k is ready to be delivered to another node while enforcing a causal order. k is the total number of events that have been seen in a node.[2]

$$P(C_k \leq t) = \prod^{i \leq k} \begin{cases} P(X_i \leq t + \delta_i) & \text{if } i \text{ is local} \\ P(R_i \leq t + \delta_i) \end{cases} \tag{4}$$

$P(R_i \leq t)$ represents the probability for a remote message to be delivered to another node. If that node is the node that issued the event, we get $P(R_i \leq t) = 1$; elsewhere it's defined according Equation 5 within an exponential latency model.

The proof for Equation 5 can be found in Section A.

$$P(R \leq t) = 1 - e^{\lambda t}/2 \tag{5}$$

[1] Such long latencies will render the system unusable when enforcing orders between events.
[2] Different values of k could be observed on different nodes at the same moment.

Equations 3 and 4 describe precisely the probability distribution for a specific message to be delivered. However, to understand the behavior of the system as a whole, and to configure mechanisms such as our reliable delivery mechanism, we need a more generic description that statistically represents all events in the system by the same probabilistic distribution.

To obtain these distributions, we first limit the number of events to be taken into account in the equation, and second, we use the rate of events to statistically represent past events.

4.3 Window of events

The main reason for limiting the number of events is that it is costly to compute the probability that all past events have been received as the number of events grows unbounded. Furthermore, the older an event is the lower impact it has in the overall probability, as the probability that it has been received tends towards one. We limit the number of events according to two factors: the visibility time of an event with a certainty degree, $t(p)$; and the global rate of events of the system, r. Equation 6 gives $t(p)$, the waiting time at which a sent message has been received by all $n-1$ other nodes with a probability of p within an exponential latency model. With this time value, we can use the rate of events to calculate the window of events $t(p).r$, which is the maximum number of previous events that the probabilistic equations should take into account.

The proof of Equation 6 in the Section A.

$$t(p) = \frac{-ln(1 - \sqrt[n-1]{p})}{\lambda} \tag{6}$$

For instance, in a system with 10 nodes and a λ of 0.3, $t(p)$ such that a message has been received by all nodes with a probability $p = 0.99$ is equal to 22.66. Assuming a rate of events $r = 30$, then we have to check the last 680 events.

4.4 Generalized FIFO

To obtain a probability distribution for every event delivery in a system enforcing FIFO, we start from Equation 3. Then, we use the time window defined above and we consider that the events are uniformly distributed between nodes : n/r events per node, where n is the number of nodes in the system and r the global rate of events. Finally, we obtain Equation 7 with $\Delta_i = i.n/r$ when considering a uniform distribution of events.

$$P(F \leq t) = \prod^{i<t(p).r/n} P(X_i \leq t + \Delta_i) \tag{7}$$

4.5 Generalized Causal

The same changes can be applied to the Equation 4. First, the number of events has to be limited. Second the time between, outgoing and incoming, events has to be generalized.

Recall that received messages, are those which arrived at a node but that cannot be seen by the application as they have missing dependencies. However, when those dependencies are fulfilled, the message is delivered.

We consider that local and received events are uniformly distributed according to the global rate. *However, we cannot consider that remote events are uniformly distributed until the event k.* Indeed, to be part of the dependency of event k a received event must be delivered. Intuitively, an event received just before has a lower chance to be delivered than an old remote event.

$P(D \leq t)$ capture this intuition and gives the probability that a received events can be delivered before t, i.e. that all the dependencies for that event have been fulfilled.[3] Note that all the information to compute this can be obtained by using information that the local node has without having to contact other nodes. $P(D \leq t)$ is obtained by counting the number of remote messages whose delay is between reception and delivery time are smaller or equal than t, and dividing it by the number of remote events, see Algorithm 1. There are two matrices in the function's header, delivery and reception. The matrices dimensions are the number of operations and the number of nodes. Then, for every operation and node they contain the time at which the operation was received or delivered, in that node.

Algorithm 1 P(D)

```
1: function P(D)(delivery, reception, t)
2:     len = nrow(delivery)
3:     n = ncol(delivery)
4:     count = 0
5:     for i = 0 to i ≤ len do
6:         for j = 0 to j ≤ n do
7:             if (delivery[i, j] − reception[i, j]) ≤ t then
8:                 count = count + 1
9:             end if
10:        end for
11:    end for
12:    return ((count − len)/((n − 1) ∗ len)
13: end function
```

To obtain Equation 8, we combine $P(D)$ with $\overline{P(R)}$, so that we get the probability "dependent and not received", and then we negate it ($\overline{\overline{P(R)}P(D)}$). We obtain $1 - (1 - P(R))P(D)$ which is the probability that a remote message is either not deliver to the node or received on the other node.

$$P(C \leq t) = \prod^{i<t(p).r} \begin{cases} P(X_i \leq t + \Delta_i) & \text{if } i = 0(mod\ n) \\ (1 - (1 - P(R_i \leq t + \Delta_i)).P(D \leq \Delta_i)) \end{cases} \tag{8}$$

The $i = 0(mod\ n)$ condition – i equals 0 modulo n – allows to distinguish local and remote events, since we consider uniform distribution of event between nodes. events. Considering a uniform distribution for the rate of events we get $\Delta_i = i/r$.

4.6 Discussion

The equations presented above describe the probabilistic behavior of messages latency in a system enforcing a given partial order. To obtain concise definitions, we made quite strong assumptions, such that a uniform distribution of

[3] In a latency model with a constant delay value, this probability should always be 1 as the events would be received in order.

events or the same latency model for every link between nodes. However, these definitions can be extended to take into account more precise system model and/or be considered as statistically enough representative to be used to configure a mechanism which is algorithmically sound as the reactive error recovery presented below.

5 Reactive Error Recovery

The Reactive Error Recovery (RER) is a mechanism to ensure a reliable message delivery. It uses the metadata required to enforce partial orders to know which messages are missing. On top of that, and in order to filter between delayed messages and lost messages, a timeout mechanism is implemented. We use our probabilistic knowledge of the network to configure the trade-off between network usage and delay time.

5.1 How does it work?

Once a message is received the system has to check if the message is deliverable since it may happen that there are previous messages that have not yet been received. Waiting before contacting the sender may be a good idea because the missing messages may arrive in between and contacting the original node may result in duplicated messages and network overuse. However if the system waits too much and the messaged was lost, then the recovery mechanism is adding an unnecessary waiting time to the recovery process. So the trade-off that we are balancing is between the network overuse versus waiting time.

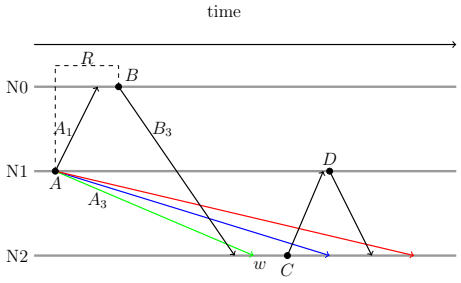


Figure 1: The three outcomes of a delayed message.

Figure 1 shows a message B which depends on message A since A was delivered to node N1 before B was issued. Therefore when the message B arrives at N3, it is seen that the message A is not there yet. At this point, a waiting time timer (w) is started, and there are three possible outcomes. First, the message is received before the waiting time's out (green arrow). This outcome would not produce additional communication since the missing element has been recovered. If the waiting time runs out, N3 contacts N2 to get a resend of the message A. However, if the original message A is received by N3 before (blue arrow) the resent message, the mechanism have produced an unnecessary request. Finally, if the original message A is received after (red line) the resent message, the mechanism successfully recovers from a missing element and obtains a delivery latency gain.

5.2 Probability of False Positives

A false positive message is defined as a message that was not lost but that the error recovery mechanism recovered, therefore duplicating a message and using unnecessary resources. An example can be seen in Figure 1, as the blue line outcome falls between message C and the return of message D.
Equation 9 gives the probability that given three events another event falls between the second and third. In the equation, there are two sets of events, A, and B, C, D. The equation gives the probability that the event A happens between the events C and D, which are issued in the following order B, C, D. And finally in order for the message A to be an FP it needs to intersect with the probability of the system being ready to deliver the message.

$$\begin{aligned} &P(C_k \leq r) \cap P(B + r + w \geq A) \\ &\cap P(B + r + w + C + D \leq A) \end{aligned} \tag{9}$$

The first part of the equation, $P(B + r + w \geq A) \cap P(B + r + w + C + D \leq A)$ can be obtained independently from the type of ordering that the system is enforcing, as it only require information for the latency model. Equation 10 gives the probability of message A arriving in between messages C and D with the exponential distribution as a latency model.

$$\begin{aligned} &P(A \geq B + r + w) \cap P(A \leq B + r + w + C + D) \\ &= \int_0^\infty \int_0^\infty \int_0^\infty \int_{b+w+r}^{b+w+r+c+d} \lambda^4 e^{-\lambda x} e^{-\lambda b} e^{-\lambda c} e^{-\lambda d} \, \mathrm{d}x \, \mathrm{d}b \, \mathrm{d}c \, \mathrm{d}d \\ &= \frac{3}{8} e^{-\lambda(r+w)} \end{aligned} \tag{10}$$

By using the defined equation 9 and setting $P(C_k \leq r)$ as the latency models CDF, $1 - e^{-\lambda r}$ if the latency model is an exponential distribution, we can then define the number of false positives that our system with have for a specific configuration.

Given a latency model with an exponential distribution and $\lambda = 1$, the FP surface function for the waiting time, w, and the time with the next event from another node once a message has arrived (time between A and B in Figure 1), r, is Equation 11.

$$f(w, r) = 3/8 * e^{-w-r} * (1 - e^{-r}) \tag{11}$$

In order to locate the highest number of false positives for a given value of r, we partially derivate equation 11 over r, and find its solution, that we will later use to find the function of false positives dependent only on the waiting time.

$$\frac{\partial f}{\partial r} 3/8 * e^{-w-r} * (1 - e^{-r}) = -3/8 * (e^r - 2) * e^{-2r-w} \tag{12}$$

If we solve the equation 12 we get its only real solution in $r = ln(2)$ and by substituting it in $f(w, r) = 3/8 * e^{-w-r} * (1 - e^{-r})$ we obtain $3/32e^{-w}$ which is the function that determines the number of false positives for a given waiting time in a system that used ordering metadata for error recovery, but didn't enforce it for message delivery.

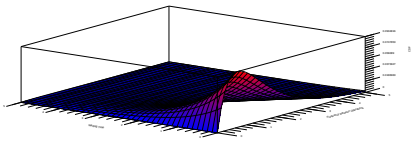


Figure 2: Surface plot of the false positive equation with exponential distribution latency model.

5.3 Eventual delivery

To ensure eventual delivery, even in the presence of faulty process, we add a retransmission timeout. After asking a resent of the missing message, we wait for a retransmission timeout (RTO), and if neither the original nor the resent message arrive, the node asks the missing to another node that knows this message (for instance N1 in Figure 1). To compute which node knows which message, we can use causal order meta-data.

Our RTO is computed using Equation 13 which is inspired from RFC6298 Section 2.2, and as our latency model is assumed constant for our scenarios.

$$RTO = SRTT + 4 \times RTTVAR \tag{13}$$

SRTT is the smoothed round trip time as it is computed as the mean round-trip time expected value of our latency model and RTTVAR is the round trip time variance, which the squared standard deviation of the latency model.

6 Related Work

The main inspiration from our work comes from the "Probabilistic Bounded Staleness" (PBS) [2], from which we show how to achieve stronger forms of consistency such as causal consistency. The PBS work is relative to the quorum literature, which includes both the deterministic approach as well as the probabilistic one. Our work is also relative to weak consistency models that require partial orders and reliable communication mechanisms.

Weak consistency levels have been formalized in order to improve the availability of systems and to overcome CAP constraints [4]. The weakest form of consistency is named eventual consistency [23]. The strongest consistency that can be achieved in a system with tolerance to network partitions while remaining available is causal+ consistency [14]. Systems like Conit and TACT [25, 26], AQuA [11], and FRACS [5], bound divergence by sacrificing availability when the constraint that bounds the divergence between nodes could be broken. TACT, provides three different metrics to measure and bound divergence. FRACS allows a delay in the propagation of updates, which sacrifices the consistency of its data in order to gain higher availability and better performance. PCAP [18], provides a probabilistic approach to solving the CAP conundrum, by providing consistency/latency trade-off SLA.

Different acknowledging mechanisms achieve reliable communication. Cumulative Acknowledgment [17], used by TCP and ABSM [21] sends back a message to the sender to notify that the last n bytes or messages have been received properly, upon the reception of this acknowledgment, the sender will know that previous messages have been received properly. In PBCF [19], a form of explicit acknowledgment, a forwarder of a message can send an acknowledgment to the interested party so that the sender does not have to. Negative Acknowledgment (or NACK) is a received-based explicit recovery approach, in which is the received the one that asks for its missing elements. Implicit Acknowledgment [15] allows the learning of reception for all those nodes that overhear a rebroadcast, and that had been waiting to rebroadcast themselves. [7] studies the probability that a message has been delivered within k steps by using a Markov chain model.

Probabilistic solutions have already been used in distributed systems. Probabilistic Quorums [16] increase the availability of the quorum systems while still ensuring a high degree of assurance. Furthermore, Probabilistic Bounded Staleness [2], studies $< k, t >$ −visibility under a WARS application model, and extends on the k-quorum systems[1] by also giving the probability that the read returns of the k-last writes. It achieves this by quantifying probabilistically the amount of staleness that a partial quorum system has given a quorum configuration (N,R,W) and a latency model (WARS and latency distribution). This gives some numbers to eventual consistency [23] and opens a new way of understanding how eventual consistency interacts with the system whilst answering two questions, (1) how eventual is EC and (2) how consistent it is.

Finally, in the reliable multicast field, pbcast [3] uses randomly generated communication trees to provide dissemination routes for its messages. Additionally, it generates a digest of the messages that is also sent to the nodes to verify missing message and request them. Another protocol, rpcast [22], is a hybrid between pbcast and lbrm [9] (a deterministic log-based reliable multicast) it order to send its messages and fallback to the deterministic solution when the probabilistic one fails. Furthermore [10] shows the probability of successfully disseminating messages, depending on the dissemination fanout, and link and node failures.

7 Conclusions

In this paper, we proposed the reactive error recovery, a reliable delivery mechanism for distributed systems requiring partial orders for their events. The configuration of this mechanism is based on a new probabilistic model for delivery latency in such systems. Our experiments show that the mechanism ensures usability of systems in the presence of lost or long-delayed message.

The cost of the mechanism can be tailored using probabilistic methods and is an order of magnitude lower than systematic acknowledgments in a broadcasting-based systems with a very high rate of communication.

This work can to extended in several directions. The probabilistic models can be enriched to fit more precisely real systems behavior, including different latency models and event distribution models. Also, an implementation of the mechanism can be done to study the behavior and the cost of the mechanism when associated to existing unreliable protocols compared to other reliable de-

livery protocols in a real-world setting.

A Appendix

We present the proof of Equation 5.

$$P(R \leq t) = 1 - e^{\lambda t}/2$$

Proof. We have two random variables named X and Y which follow an exponential distribution with the same mean value.The pdf[4] for each one is $\lambda e^{-\lambda x}$ and $\lambda e^{-\lambda y}$. Given that they are independent, we can define the combined pdf between X and Y as: $\lambda^2 e^{-\lambda x} e^{-\lambda y}$.

As $P(R \leq t)$ is the cdf[5] of the combined pdf, then:

$$P(R \leq t) = P(X - Y \leq t) = \iint \lambda^2 e^{-\lambda x} e^{-\lambda y} \, \mathrm{d}x \, \mathrm{d}y$$

For $Y \leq X + t$:

$$P(R \leq t) = \lambda^2 \int_0^{\infty} e^{-\lambda x} \left(\int_0^{x+t} e^{-\lambda y} \right) \mathrm{d}y \, \mathrm{d}x$$

By solving the first integral we get:

$$P(R \leq t) = \lambda \int_0^{\infty} (e^{-\lambda x} - e^{-\lambda t} e^{-2\lambda x}) \, \mathrm{d}x$$

That results in:

$$P(R \leq t) = \lambda \left(\frac{1}{\lambda} - \frac{e^{-\lambda t}}{2\lambda} \right) = 1 - 0.5 e^{-\lambda t}$$

□

This is the proof for Equation 6.

$$t(p) = \frac{-ln(1 - \sqrt[n-1]{p})}{\lambda}$$

Proof. $t(p)$ is the quantile function for the exponential distribution, in which $n-1$ is the number of nodes to which a message is send. Therefore after time $t(p)$ there is a p probability that the message has been received by all $n-1$ nodes.

The quantile function for the exponential distribution is $Q(p) = \frac{-ln(1-p)}{\lambda}$.

The probability that a message is received by all $n-1$ nodes is defined as $P(X_g \leq t)$ as is equal to the $n-1$ probabilities that a message is received by a node multiplied, $P(X_l \leq t)^{n-1}$.

[4] PDF:Probability Density Function
[5] CDF:Cumulative Distribution Function

If we want $p = P(X_g \leq t(p))$ then $p = \sqrt[n-1]{P(X_l \leq t(p))}$

Combining the probability that a message is received by all $n-1$ nodes and the quantile function, we get: $t(p) = \frac{-ln(1-\sqrt[n-1]{p})}{\lambda}$ □

References

[1] Amitanand S. Aiyer, Lorenzo Alvisi, and Rida A. Bazzi. Byzantine and multi-writer k-quorums. In *Proceedings of the 20th International Conference on Distributed Computing*, DISC'06, pages 443–458, Berlin, Heidelberg, 2006. Springer-Verlag. URL: http://dx.doi.org/10.1007/11864219_31, doi:10.1007/11864219_31.

[2] Peter Bailis, Shivaram Venkataraman, Michael J. Franklin, Joseph M. Hellerstein, and Ion Stoica. Probabilistically bounded staleness for practical partial quorums. *Proc. VLDB Endow.*, 5(8):776–787, April 2012. URL: http://dx.doi.org/10.14778/2212351.2212359, doi:10.14778/2212351.2212359.

[3] Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2):41–88, May 1999. URL: http://doi.acm.org/10.1145/312203.312207, doi:10.1145/312203.312207.

[4] E.A. Brewer. Towards robust distributed systems. In *Proceedings of the Annual ACM Symposium on Principles of Distributed Computing*, volume 19, pages 7–10, 2000.

[5] Chi Zhang and Zheng Zhang. Trading replication consistency for performance and availability: an adaptive approach. In *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, volume 2, pages 687–695. IEEE, 2003. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1203520, doi:10.1109/ICDCS.2003.1203520.

[6] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. Spanner: Google's globally distributed database. *ACM Trans. Comput. Syst.*, 31(3):8:1–8:22, August 2013. URL: http://doi.acm.org/10.1145/2491245, doi:10.1145/2491245.

[7] Mozhdeh Gholibeigi, Geert Heijenk, Dmitri Moltchanov, and Yevgeni Koucheryavy. Analysis of a receiver-based reliable broadcast approach for vehicular networks. *Ad Hoc Netw.*, 37(P1):63–75, February 2016. URL: http://dx.doi.org/10.1016/j.adhoc.2015.08.003, doi:10.1016/j.adhoc.2015.08.003.

[8] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33:51–59, June 2002.

[9] Hugh W. Holbrook, Sandeep K. Singhal, and David R. Cheriton. Log-based receiver-reliable multicast for distributed interactive simulation. *SIGCOMM Comput. Commun. Rev.*, 25(4):328–341, October 1995. URL: http://doi.acm.org/10.1145/217391.217468, doi:10.1145/217391.217468.

[10] Anne-Marie Kermarrec, Laurent Massoulié, and Ayalvadi J. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Trans. Parallel Distrib. Syst.*, 14(3):248–258, March 2003. URL: http://dx.doi.org/10.1109/TPDS.2003.1189583, doi:10.1109/TPDS.2003.1189583.

[11] Sudha Krishnamurthy, W.H. Sanders, and Michel Cukier. An adaptive quality of service aware middleware for replicated services. *IEEE Transactions on Parallel and Distributed Systems*, 14(11):1112–1125, nov 2003. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1247672, doi:10.1109/TPDS.2003.1247672.

[12] Avinash Lakshman and Prashant Malik. Cassandra - a decentralized structured storage system. In *Proceedings of The 3rd ACM SIGOPS International Workshop on Large Scale Distributed Systems and Middleware (LADIS)*, Big Sky, MT, USA, October 2009. SIGOPS.

[13] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.

[14] Wyatt Lloyd, Michael J. Freedman, Michael Kaminsky, and David G. Andersen. Don't settle for eventual: scalable causal consistency for wide-area storage with cops. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, SOSP '11, pages 401–416, New York, NY, USA, 2011. ACM.

[15] Xiaomin Ma, Jinsong Zhang, Xiaoyan Yin, and Kishor S Trivedi. Design and analysis of a robust broadcast scheme for vanet safety-related services. *IEEE Transactions on Vehicular Technology*, 61(1):46–61, 2012.

[16] Dahlia Malkhi, Michael K Reiter, Avishai Wool, and Rebecca N Wright. Probabilistic Quorum Systems. *Information and Computation*, 170(2):184–206, nov 2001. URL: http://linkinghub.elsevier.com/retrieve/pii/S0890540101930548, doi:10.1006/inco.2001.3054.

[17] DARPA Internet Program. Transmission Control Protocol. RFC 793, RFC Editor, September 1981. URL: https://tools.ietf.org/rfc/rfc793.txt.

[18] Muntasir Raihan Rahman, Lewis Tseng, Son Nguyen, Indranil Gupta, and Nitin H. Vaidya. Characterizing and adapting the consistency-latency tradeoff in distributed key-value stores. *CoRR*, abs/1509.02464, 2015. URL: http://arxiv.org/abs/1509.02464.

[19] Rajapandiyan Rajendran and Jan De Jongh. An efficient and reliable multi-hop geographical broadcast protocol in vehicular ad-hoc networks. In *ITS Telecommunications (ITST), 2013 13th International Conference on*, pages 1–7. IEEE, 2013.

[20] Matthias Ressel, Doris Nitsche-Ruhland, and Rul Gunzenhäuser. An integrating, transformation-oriented approach to concurrency control and undo in group editors. In *Conference on Computer supported cooperative work (CSCW)*, pages 288–297, Boston, MA, USA, 1996.

[21] Francisco J. Ros, Pedro M. Ruiz, and Ivan Stojmenovic. Acknowledgment-based broadcast protocol for reliable and efficient data dissemination in vehicular ad hoc networks. *IEEE Transactions on Mobile Computing*, 11(1):33–46, January 2012. URL: http://dx.doi.org/10.1109/TMC.2010.253, doi:10.1109/TMC.2010.253.

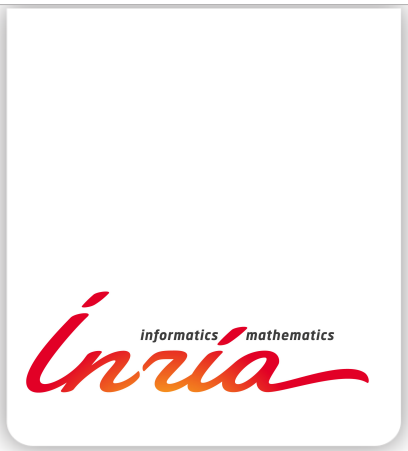
[22] Qixiang Sun and Daniel C. Sturman. A gossip-based reliable multicast for large-scale high-throughput applications. In *Proceedings of the 2000 International Conference on Dependable Systems and Networks (Formerly FTCS-30 and DCCA-8)*, DSN '00, pages 347–, Washington, DC, USA, 2000. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=647881.737931.

[23] Werner Vogels. Eventually consistent. *Commun. ACM*, 52(1):40–44, January 2009.

[24] Stéphane Weiss, Pascal Urso, and Pascal Molli. Logoot: A scalable optimistic replication algorithm for collaborative editing on P2P networks. In *29th IEEE International Conference on Distributed Computing Systems (ICDCS 2009)*, pages 404 –412, Montréal, QC, Canada, jun. 2009. IEEE Computer Society.

[25] Haifeng Yu and Amin Vahdat. The costs and limits of availability for replicated services. *ACM SIGOPS Operating Systems Review*, 35(5):29, dec 2001. URL: http://portal.acm.org/citation.cfm?doid=502059.502038, doi:10.1145/502059.502038.

[26] Haifeng Yu and Amin Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Transactions on Computer Systems*, 20(3):239–282, aug 2002. URL: http://portal.acm.org/citation.cfm?doid=566340.566342, doi:10.1145/566340.566342.



RESEARCH CENTRE
NANCY – GRAND EST

615 rue du Jardin Botanique
CS20101
54603 Villers-lès-Nancy Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399